

⑯ BUNDESREPUBLIK  
DEUTSCHLAND



DEUTSCHES  
PATENTAMT

⑯ Offenlegungsschrift  
⑯ DE 197 35 981 A 1

⑯ Int. Cl. 6:  
G 06 F 9/38  
G 06 F 12/08  
G 06 F 15/76  
// H04N 7/26

⑯ Aktenzeichen: 197 35 981.7  
⑯ Anmeldetag: 19. 8. 97  
⑯ Offenlegungstag: 26. 3. 98

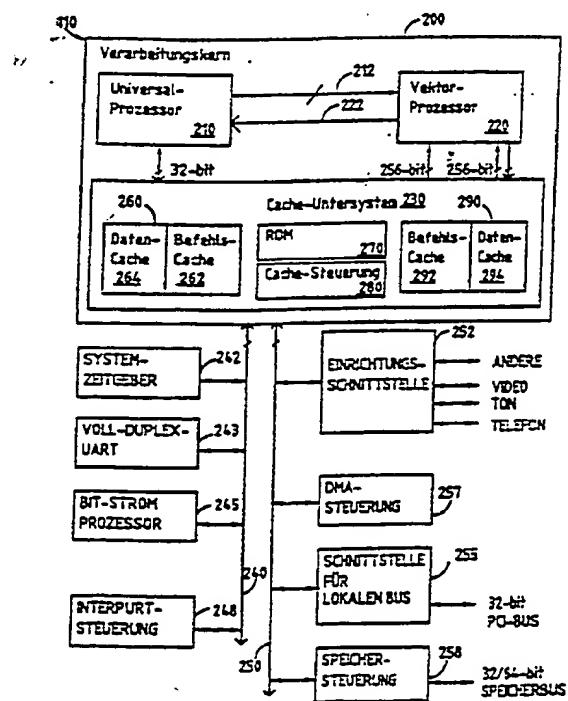
⑯ Unionspriorität:  
697102 19.08.96 US

⑯ Anmelder:  
Samsung Electronics Co., Ltd., Suwon, Kyungki, KR.  
⑯ Vertreter:  
Kahler, Käck & Fiener, 86899 Landsberg

⑯ Erfinder:  
Nguyen, Le Trong, Monte Serena, Calif., US

⑯ Mehrprozessorbetrieb in einem Multimedia-Signalprozessor

⑯ Um eine hohe Leistungsfähigkeit bei niedrigen Kosten zu erzielen, verwendet ein integrierter Digitalprozessor eine Architektur, die sowohl einen Universal-Prozessor (210) als auch einen Vektorprozessor (220) umfaßt. Der integrierte Digitalsignalprozessor umfaßt auch ein Cache-Untersystem (230), einen ersten Bus und einen zweiten Bus. Das Cache-Untersystem (230) bietet eine Cachespeicherung und eine Datenleitung für die Prozessoren und Busse. Mehrere simultan einrichtbare Kommunikationsbahnen können für die Prozessoren und Busse in dem Cache-Untersystem verwendet werden. Ferner werden in dem Cache-Untersystem gleichzeitige Lese- und Schreibvorgänge auf einem Cache-Speicher unterstützt.



## Beschreibung

Die Erfindung bezieht sich auf **Digital-Signal-Prozessoren** und insbesondere auf **asymmetrische Parallel-Dual-Teilprozessingssysteme**, die einen **Universal-Prozessor** und einen **Vektorprozessor** zum Handhaben von **Vektor-daten** umfassen.

Eine Verschiedenheit von **Digital-Signal-Prozessoren (DSPs)** wird bei **Multimediaanwendungen** verwendet, beispielsweise dem Codieren und Decodieren von Bild-, Ton- und Kommunikationsdaten. Ein Typ eines **Digital-Signalprozessors (DSP)** weist eine fest zugeordnete Hardware zum Ansprechen eines bestimmten Problems auf, beispielsweise einer **MPEG-Bild- bzw. MPEG-Video-Decodierung** oder -**Codierung**. **DSPs mit fest zugeordneter Hardware** bieten mit Blick auf die Kosten im allgemeinen eine hohe Leistungsfähigkeit, sind aber nur für bestimmte Probleme verwendbar und nicht in der Lage, an andere Probleme angepaßt zu werden oder hinsichtlich Standards geändert zu werden.

**Programmierbare DSPs** führen Programme aus, die **Multimediacomprobleme** lösen, und bieten eine größere **Flexibilität** als **DSPs mit fest zugeordneter Hardware**, da das Ändern der Software für einen programmierbaren **DSP** das gelöste Problem ändern kann. Ein Nachteil programmierbarer **DSPs** ist deren geringe Leistungsfähigkeit mit Blick auf die Kosten. Ein programmierbarer **DSP** weist typischerweise einen Aufbau ähnlich dem eines **Universal-Prozessors** auf und hat eine relativ geringe **Verarbeitungs- bzw. Prozessingleistung**. Die geringe **Verarbeitungsleistung** folgt im allgemeinen aus einem Versuch, die Kosten zu minimieren. Daher ist ein solcher **DSP** nicht vollständig zufriedenstellend, da ein **DSP** mit geringer Leistung die Möglichkeit des **DSP** hindert, **komplexere Multimediacomprobleme** anzusprechen, wie beispielsweise eine **Echtzeit-Bildcodierung** und -**decodierung**.

Da ein Ziel für programmierbare **DSPs** darin besteht, eine hohe **Verarbeitungsleistung** zum Ansprechen bzw. **Adressieren** von **Multimediacomproblemen** bei **minimalen Kosten** bereit zu stellen, könnte man in einen solchen **DSP** eine **Parallelverarbeitung** aufnehmen, was ein bekannter Weg zum Erhöhen der **Verarbeitungsleistung** ist. Eine **Architektur** für eine **Parallelverarbeitung** hat ein "very long instruction word"-(VLIW)-**DSP** bzw. **DSP** für sehr lange Befehlsworte, der durch eine große Anzahl von **Funktionseinheiten** gekennzeichnet ist, von denen die meisten verschiedene aber relativ einfache Aufgaben ausführen. Eine einzelne Anweisung bzw. ein einzelner Befehl für einen **VLIW-DSP** kann 128 Byte lang oder länger sein und weist getrennte Teile auf. Jedes Teil kann durch getrennte **Funktionseinheiten** parallel ausgeführt werden. **VLIW-DSP** weisen eine sehr hohe **Berechnungs- bzw. Verarbeitungsleistung** auf, da eine große Anzahl von **Funktionseinheiten** parallel betrieben werden kann. **VLIW-DSP** sind auch relativ kostengünstig, da jede **Funktionseinheit** relativ klein und einfach aufgebaut ist. Ein Problem hinsichtlich **VLIW-DSPs** ist jedoch die schlechte Leistungsfähigkeit beim Handhaben der **Eingabe-/ Ausgabesteuerung**, der **Kommunikation** mit einem **Haupt- bzw. Hostcomputer** und anderer **Funktionen**, die sich nicht selber die Fähigkeit zu einer parallelen Ausführung in den **Funktionseinheiten** des **VLIW-DSP** verleihen. Zudem unterscheiden sich Programme für **VLIW** von konventionellen Computerprogrammen und können schwierig zu entwickeln sein, da es an **Programmierwerkzeugen** und **Programmierern** mangelt, die mit **VLIW-Softwarearchitekturen** vertraut sind.

Die Aufgabe der vorliegenden Erfindung besteht darin, die **Ineffizienz** bei **Digital-Signalprozessoren** und insbesondere **asymmetrischen Parallel-Dual-Teilprozess-Systemen** zu verringern.

Diese Aufgabe wird durch einen Prozessor mit den Merkmalen gemäß Anspruch 1 gelöst.  
Vorteilhafte Ausgestaltungen sind Gegenstand von Unteransprüchen.

Der **Digital-Signalprozessor** kombiniert insbesondere einen **Universal-Prozessor** mit einem **Vektorprozessor**, der parallel zu dem **Universal-Prozessor** betrieben werden kann. Der integrierte **Digital-Signal-Prozessor** ist in der Lage, eine hohe Leistungsfähigkeit bei niedrigen Kosten zu erzielen, da die beiden Prozessoren nur Aufgaben ausführen, die für jeden Prozessor ideal liegen. Z. B. läßt der **Universal-Prozessor** ein **Echtzeit-Betriebssystem** laufen und führt eine **Gesamtsystemhandhabung** durch, während der **Vektorprozessor** zum Durchführen von **Parallelberechnungen** unter Verwendung von **Datenstrukturen** verwendet wird, die als "Vektoren" bezeichnet werden. Ein Vektor ist eine Sammlung von **Datenelementen**, die typischerweise vom gleichen Typ sind.

Bei einem Ausführungsbeispiel umfaßt der **Digital-Signalprozessor** auch ein **Cache-Subsystem** bzw. **Cache-Untersystem**, einen ersten Bus und einen zweiten Bus. Der erste Bus wird für **Hochgeschwindigkeitseinrichtungen**, beispielsweise eine lokale Busschnittstelle, eine **DMA-Steuereinrichtung**, eine **Einheitensteuereinrichtung** und eine **Speichersteuereinrichtung** verwendet. Der zweite Bus wird für **Einrichtungen** mit einer geringen Geschwindigkeit verwendet, beispielsweise einen **Systemzeitgeber**, eine **UAPT** (Anpassungsschaltung zur Umsetzung von paralleler zu serieller Datenübertragung), einen **Bitstrom-Prozessor** und eine **Interrupt- bzw. Unterbrechungs-Steuereinheit**.

Das **Cache-Subsystem** kombiniert **Cachefunktionen** mit **Vermittlungs- oder Datenführungsfunctionen**. Die **Vermittlungsfunktionen** ermöglichen mehrere **Kommunikationswege** zwischen den Prozessoren und Bussen, um gleichzeitig betrieben werden zu können. Ferner ermöglicht das **Cacheteil** des **Cache-Subsystems** ein gleichzeitiges **Lesen** und **Schreiben** im **Cachespeicher**.

Die Erfindung wird nachstehend beispielsweise anhand der Zeichnung näher erläutert. Es zeigen:

- Fig. 1 ein Blockdiagramm einer **Multimediacarte** gemäß einem Ausführungsbeispiel;
- Fig. 2 ein Blockdiagramm eines **Multimedia-Signalprozessors** gemäß einem Ausführungsbeispiel;
- Fig. 3 Beziehungen zwischen Prozessoren und Software oder Firmware in einem System, das einen **Multimediacoprozessor** gemäß einem Ausführungsbeispiel umfaßt;
- Fig. 4 ein Blockdiagramm eines **Cache-Untersystems** gemäß einem Ausführungsbeispiel;
- Fig. 5 ein **Speicherabbild** bzw. einen **Adressenumsetzer** gemäß einem Ausführungsbeispiel;
- Fig. 6 ein Blockdiagramm einer **Datenpipeline** bzw. **Datenleitung**, die bei einem **Cache-Untersystem** gemäß einem Ausführungsbeispiel verwendet wird;

Fig. 7 ein Blockdiagramm einer zweiten Datenpipeline, die bei einem Cache-Untersystem gemäß einem Ausführungsbeispiel verwendet wird; und

Fig. 8 ein Blockdiagramm einer Adresspipeline, die bei einem Cache-Untersystem gemäß einem Ausführungsbeispiel verwendet wird.

Das Verwenden der gleichen Bezugssymbole in verschiedenen Figuren zeigt ähnliche oder identische Merkmale.

Gemäß einem Aspekt der Erfindung umfaßt ein Multimediacoprozessor einen Universal-Prozessor bzw. einen Prozessor für einen allgemeinen Zweck und einen Vektorprozessor, der entsprechend zum Trennen von Programm-Teilprozessen bzw. einem gereihten Programmcode parallel betrieben wird. Der Universal-Prozessor führt wie die meisten konventionellen Prozessoren für allgemeine Anwendungen Anweisungen bzw. Befehle aus, die typischerweise skalare Daten manipulieren bzw. handhaben. Solche Prozessoren sind zum Ausführen von Eingabe-/Ausgabe-(I/O)- und Steuerfunktionen geeignet. Bei einigen Ausführungsbeispielen weist der Universal-Prozessor eine begrenzte Vektorverarbeitungsfähigkeit von Datenelementen mit einer Größe von mehreren Byte auf, die in ein Datenwort gepackt sind. Falls der Universal-Prozessor z. B. ein 32-Bit-Prozessor ist, kann der Universal-Prozessor gemäß einiger Ausführungsbeispiele vier Ein-Byte-Datenelemente simultan verarbeiten. Jedoch macht eine Multimediacberechnung bzw. Multimediasverarbeitung, wie beispielsweise eine Ton- und Bild-Datenkompression und -dekompression viele wiederholte Berechnungen hinsichtlich Bildpunktanordnungen und Ketten aus Tondaten erforderlich. Um Realzeit- bzw. Echtzeit-Multimediaoperationen auszuführen, muß ein Universal-Prozessor der skalare Daten (z. B. einen Bildpunktwert oder eine Tonamplitude pro Operand) oder nur kleine Vektoren handhabt, mit einer hohen Taktfrequenz betrieben werden. Im Gegensatz dazu führt der Vektorprozessor Anweisungen aus, wo jeder Operand ein Vektor ist, der mehrere Datenelemente enthält (z. B. mehrere Bildpunktswerte oder Tonamplituden). Daher kann der Vektorprozessor Echtzeit-Multimediaoperationen mit einem Bruchteil der Taktfrequenz durchführen, die für einen Universal-Prozessor erforderlich ist, um die gleiche Funktion auszuführen.

So bietet durch das Ermöglichen einer effizienten Teilung der Aufgaben, die für eine Multimediaanwendung erforderlich sind, die Kombination aus einem programmierbaren Universal-Prozessor und einem Vektorprozessor mit Blick auf die Kosten eine hohe Leistungsfähigkeit. Bei einem Ausführungsbeispiel führt der Universal-Prozessor ein Echtzeit-Betriebssystem aus, das für eine Medien-Leiterplatte ("Karte") entwickelt wurde, das mit einem Host- bzw. Haupt-Computersystem kommuniziert. Das Echtzeit-Betriebssystem kommuniziert mit einem primären Prozessor des Computersystems, stellt I/O- bzw. EIN/AUS-Einrichtungen ein oder koppelt sie mit der Karte und wählt Aufgaben, die der Vektorprozessor ausführt. Bei diesem Ausführungsbeispiel ist der Vektorprozessor entworfen worden, die berechnungsmäßig intensiven Aufgaben durchzuführen, wie eine Handhabung großer Datenblöcke erforderlich machen, während der Universal-Prozessor als der Haupt- bzw. Master-Prozessor für den Vektorprozessor dient. Programmteilprozesse bzw. gereihte Codeabschnitte eines Programms werden für jeden Prozessor unter Verwendung eines konventionellen Befehlssatzes bzw. Anweisungssatzes geschrieben, was den Multimediacoprozessor "programmiererfreundlich" macht. Die Programmierbarkeit ermöglicht, daß der Multimediacoprozessor eine Vielzahl verschiedener Multimediaaufgaben ausführen kann. Der Multimediacoprozessor kann z. B. an ein neues Protokoll einfach dadurch angepaßt werden, daß entweder seine Anwendungsprogramme oder seine Firmware geändert werden. Bei einem Ausführungsbeispiel ist der Anweisungs- bzw. Befehlssatz ähnlich dem eines konventionellen RISC-Anweisungssatzes (RISC steht für "reduced instruction set computer" bzw. Computer mit reduziertem Anweisungssatz).

Gemäß einem anderen Aspekt teilen der Universal-Prozessor und der Vektorprozessor eine Vielzahl an verschiedenen auf dem Chip befindlichen und vom Chip getrennten Ressourcen bzw. Quellen, die durch einen einzelnen Adressraum zugreifbar sind. Ein Cache-Untersystem bzw. -Subsystem, das getrennte Daten- und Anweisungs-Caches für jeden Prozessor implementiert, bietet auch eine Verbindung vom Schaltplattentyp bzw. Schaltkartentyp zwischen einem lokalen Speicher und Ressourcen, wie beispielsweise einem Bitstrom-Prozessor, einer universellen, asynchronen Empfänger-Sendereinheit ("UART"), einem Direktzugriffsspeicher-(DMA)-Kontroller bzw. einer Direktzugriffsspeicher-Steuereinrichtung, einer Schnittstelle für einen lokalen Bus bzw. einer lokalen Busschnittstelle und einer Codierer-Decodierer-(“CODEC”)-Schnittstelle, die speicherabgebildete Einrichtungen darstellen. Das Cache-Untersystem kann ein Transaktions-orientiertes Protokoll verwenden, das eine Zentrale bzw. Schaltplatte für einen Datenzugriff auf die einzelnen Prozessoren und speicherabgebildeten Ressourcen implementiert bzw. umsetzt.

Fig. 1 stellt eine Multimediacarte 100 gemäß einem Ausführungsbeispiel dar. Die Multimediacarte 100 umfaßt eine Leiterplatte, einen Multimediacoprozessor 110 und eine Verbindungseinrichtung, die an einem lokalen Bus 105 eines Host-Computersystems befestigt ist. Bei einem beispielhaften Ausführungsbeispiel ist der lokale Bus 105 ein PCI-Bus; jedoch kann der lokale Bus 105 bei anderen Ausführungsbeispielen ein Marken-Bus oder ein Bus sein, der an irgendein gewünschtes Protokoll angepaßt ist, wie beispielsweise die ISA- oder VESA-Busprotokolle.

Der Multimediacoprozessor 110 verwendet einen lokalen Speicher 120, der auch auf der Multimediacarte 100 angeordnet ist, zum Speichern von Daten- und Programmanweisungen. Der lokale Speicher 120 kann auch als ein Frame- bzw. Vollbildpuffer für Bild- bzw. Videocodierungsanwendungen und Bild-Decodierungsanwendungen dienen. Beim beispielhaften Ausführungsbeispiel kann der lokale Speicher 120 durch einen synchronen, dynamischen 512K x 32-Bit-Direktzugriffsspeicher (DRAM) realisiert bzw. implementiert werden. Teile des lokalen Speicherraums können auch durch einen statischen Einchip-Direktzugriffsspeicher ("SRAM") und einen Nurlesespeicher ("ROM") im Multimediacoprozessor 110 implementiert werden. Tatsächlich braucht der lokale Speicher 120 nicht implementiert zu werden, falls ein Einchip-Speicher vorgesehen wird, um die Daten und Anweisungen der Multimediacarte 100 in ausreichendem Maße zu halten.

Zusätzlich zum Multimediacoprozessor 110 und zum lokalen Speicher 120 umfaßt die Multimediacarte 100

5 einen Bildanalog-zu-digital-Wandler (ADC) 132, einen Bild-digitalzu-analog-Wandler (DAC) 134, einen Ton-ADC 142, einen Ton-DAC 144, einen Kommunikations-ADC 146 und einen Kommunikations-DAC 148. Jeder der Wandler 132, 134, 142, 144, 146 und 148 kann durch eine oder mehrere getrennte integrierte Schaltungen implementiert werden. Alternativ können zwei oder mehr der Wandler 132, 134, 142, 144, 146 und 148 integriert auf einer einzelnen integrierten Schaltung vorgesehen werden. Z. B. kann eine einzelne integrierte Schaltung 140, beispielsweise die AD1843, die bei Analog Devices, Inc. verfügbar ist, die Funktionen der Wandler 142, 144, 146 und 148 implementieren.

10 Der Bild- bzw. Video-ADC 132, der beispielsweise durch eine integrierte KS0122-Schaltung implementiert werden kann, die von Samsung Semiconductor, Inc. erhältlich ist, stellt eine Verbindung zu einer Videokamera oder einer anderen Quelle eines Bildsignals dar und digitalisiert das Video- oder Bildsignal in eine Reihe von 15 Bildpunktwerten. Die Multimediacarte 100 komprimiert oder codiert die Bildpunktwerthe gemäß einem Bildcodierungsstandard, beispielsweise MPEG, JPEG oder H.264, der in der Firmware implementiert ist, die durch den Multimedia-Prozessor 110 ausgeführt wird. Die codierten Bilddaten können dann über einen lokalen Bus 105 zum Host-Computer übertragen werden, und zwar zu einer Einrichtung, beispielsweise einer Ethernetkarte, die mit einem lokalen Bus 105 gekoppelt ist, oder für eine Übertragung auf einer Telefonleitung, die mit einem Kommunikations-DAC 148 gekoppelt ist, weiter codiert werden.

20 Der Bild-DAC 134 wandelt eine Reihe von digitalen Abtastwerten vom Multimedia-Prozessor 110 in ein analoges Bildsignal bzw. Videosignal für einen Bildmonitor oder ein Fernsehgerät um. Der Bild-DAC 134 kann z. B. durch eine integrierte KS0119-Schaltung, die bei Samsung Semiconductor, Inc. erhältlich ist, gemäß einem NTSC- oder PAL-Bildstandard implementiert werden. Der Multimedia-Prozessor 110 kann die Reihe digitaler 25 Abtastwerte für den Bild-DAC 134 aufgrund von Daten erzeugen, die von dem Host-Computer, einer anderen Einrichtung, die mit dem lokalen Bus 105 gekoppelt ist, einer Videokamera, die mit dem Bild-ADC 132 gekoppelt ist oder einer Telefonleitung empfangen werden, die mit dem Kommunikations-ADC 146 gekoppelt ist.

30 Eine optionale Komponente der Multimediacarte 100 ist eine Grafik-Steuereinrichtung 150, die den lokalen Speicher 120 mit dem Multimedia-Prozessor 110 teilt und ein Bildsignal für einen Videomonitor für das Hostsystem bietet. Die Grafiksteuereinrichtung 150 kann z. B. durch eine Super-VGA-Grafiksteuereinrichtung implementiert werden, die von verschiedenen Händlern und Herstellern, beispielsweise Cirrus Logic, S3 oder Trident Microsystems erhältlich ist. Der Multimedia-Prozessor 110 erzeugt Bildpunktabbildungen in einem lokalen Speicher 120 und speichert sie dort, wovon die Grafik-Steuereinrichtung 150 ein Bildsignal für den Videomonitor 35 erzeugt.

35 Der Ton- bzw. Audio-ADC 142 und der Audio- bzw. Ton-DAC 144 sind zum Eingeben und Ausgeben analoger Tonsignale. Gemäß einem Aspekt emuliert die Multimediacarte 100 die Funktionen einer Ton- bzw. Soundkarte, beispielsweise der populären "SoundBlaster", und implementiert Tonsynthesefunktionen, wie beispielsweise eine Wavetablebzw. Wellentabellensynthese und eine FM-Synthese. Für Tonkartenemulationen bzw. Soundkartenemulationen stellt ein Anwendungsprogramm, das durch den Hostcomputer ausgeführt wird, Daten bereit, die einen Ton darstellen, und der Multimedia-Prozessor 110 erzeugt Tonamplituden entsprechend dieser Daten. Der Ton-DAC 144 wandelt die Tonamplituden für einen Lautsprecher oder Verstärker in ein analoges Tonsignal um. Der Multimedia-Prozessor 110 handhabt in ähnlicher Weise eingegebene Tonsignale vom Ton-ADC 142.

40 Der Kommunikations-ADC 146 tastet ein analoges Signal von einer Telefonleitung ab und bietet digitalisierte Abtastwerte für den Multimedia-Prozessor 110. Wie der Multimedia-Prozessor 110 die digitalisierten Abtastwerte verarbeitet, hängt von der Funktion ab, die in der Firmware implementiert ist. Z. B. kann der Multimedia-Prozessor 110 Modemfunktionen durch Ausführungsprogramme in Firmware implementieren, die eine V.34-De-45 modulation der Abtastwerte und eine V.42 bis Fehlerkorrektur und Dekomprimierung durchführen. Der Multimedia-Prozessor 110 kann auch Daten komprimieren, die vom Hostcomputer empfangen werden und digitale Abtastwerte zum Darstellen eines korrekt modulierten analogen Signals zum Übertragen durch den Kommunikations-DAC 148 erzeugen. Ähnlich kann der Multimedia-Prozessor 110 eine Antwortmaschine, Fax oder Bildtelefon-Funktionen unter Verwendung der gleichen Hardware (ADC 146 und DAC 148) als der Schnittstelle 50 zu den Telefonleitungen implementieren, falls eine geeignete Firmware oder Software verfügbar ist.

55 Fig. 2 stellt ein Blockdiagramm eines Ausführungsbeispiels des Multimedia-Prozessors 110 dar. Der Multimedia-Prozessor 110 umfaßt einen Verarbeitungskern bzw. Prozessingkern 200, der einen Universal-Prozessor 210 und einen Vektorprozessor 220 enthält. Wie er hier verwendet wird, bezieht sich der Ausdruck Vektorprozessor auf einen Prozessor, der Anweisungen mit Vektoroperanden ausführt, d. h. Operanden, die jeweils mehrere Datenelemente des gleichen Typs enthalten. Jeder, der Universal-Prozessor 210 und der Vektorprozessor 220 führt einen getrennten Programmteilcode aus und kann ein Skalarrechner bzw. Skalarprozessor oder ein Superskalarprozessor sein.

60 Bei dem beispielhaften Ausführungsbeispiel ist der Universal-Prozessor 210 ein 32-Bit-RISC-Prozessor, der mit 40 MHz betrieben wird und dem Standard-ARM7-Anweisungssatz angepaßt ist. Die Architektur für einen ARM7-RISC-Prozessor und den ARM7-Anweisungssatz ist in dem ARM7DM-Datenblatt beschrieben, das von Advanced RISC Machines Ltd. erhältlich ist. Der Universal-Prozessor 210 implementiert auch eine Erweiterung des ARM7-Anweisungssatzes, die Anweisungen bzw. Befehle für eine Schnittstelle mit dem Vektorprozessor 220 umfaßt. Die gemeinsam anhängige Patentanmeldung mit dem Titel "System and Method for Handling Software Interrupts with Argument Passing", die durch Bezugnahme aufgenommen wird, beschreibt die Erweiterung des ARM7-Anweisungssatzes für das beispielhafte Ausführungsbeispiel der Erfindung. Der Universal-Prozessor 210 ist über einen Steuerbus 212 mit dem Vektorprozessor 220 verbunden, um die Erweiterung des 65 ARM7-Anweisungssatzes auszuführen. Ferner wird eine Unterbrechungsleitung bzw. Interruptleitung 222 durch den Vektorprozessor 220 verwendet, um eine Unterbrechung bzw. einen Interrupt am Universal-Prozessor 210 abzufragen.

Der Vektorprozessor 220 weist eine SIMD-(Einzelanweisungsmehrfaeche Daten)-Architektur auf und handhabt sowohl skalare als auch vektorielle Großen. Beim beispielhaften Ausführungsbeispiel besteht der Vektorprozessor 220 aus einer RISC-Zentralverarbeitungseinheit bzw. RISC-Zentraleinheit im Pipelinebetrieb, die mit 80 MHz betrieben wird und eine 288-Bit-Vektoraufzeichnungsdatei bzw. -Vektorregisterdatei aufweist. Jedes Vektorregister in der Vektorregisterdatei kann bis zu 32 Datenelemente enthalten. Tabelle 1 stellt die Datentypen dar, die für Datenelemente innerhalb eines Vektors gestützt werden.

Tabelle 1

Datentyp	Datengröße	Interpretation
int8	8 Bit (Byte)	8-Bit-2'er Komplement-Ganzzahl zwischen -128 und 127
int9	9 Bit (Byte9)	9-Bit-2'er Komplement-Ganzzahl zwischen -256 und 255
int16	16 Bit (Halbwort)	16-Bit-2'er Komplement zwischen -32.768 und 32.767.
int32	32 Bit (Wort)	32-Bit-2'er Komplement-Ganzzahl zwischen -2147483648 und 2147483647
Gleit- punkt	32 Bit (Wort)	32-Bit-Gleitpunktzahl, die sich an das IEEE-754-Single- bzw. Einzel-Präzisionsformat anpaßt.

Daher kann ein Vektorregister zweiunddreißig 8-Bit- oder 9-Bit-Ganzzahl-Datenelemente, sechzehn 16-Bit-Ganzzahl-Datenelemente oder acht 32-Bit-Ganzzahl- oder Gleitpunktelemente halten. Zudem kann das beispielhafte Ausführungsbeispiel auch mit einem 576-Bit-Vektor-Operanden arbeiten, der zwei Vektorregister aufspannt.

Der Anweisungssatz für den Vektorprozessor 220 umfaßt Anweisungen zum Handhaben von Vektoren und zum Manipulieren von Skalaren. Die Patentanmeldung mit dem Titel "Einzelbefehl-Mehrdaten-Verarbeitung bei einem Multimedia-Signalprozessor", die durch Bezugnahme eingeschlossen wird, beschreibt den Anweisungssatz für das beispielhafte Ausführungsbeispiel und einen Aufbau bzw. eine Architektur zum Implementieren des Anweisungssatzes.

Das Cache-Untersystem 230 enthält einen SRAM-Block 260, der graphisch als zwei Blöcke dargestellt ist, ein ROM 270 und eine Cache-Steuereinrichtung 280. Das Cache-Untersystem 230 kann den SRAM-Block 260 in (i) einen Anweisungs-Cache 262 und einen Daten-Cache 264 für den Universal-Prozessor 210 und (ii) einen Anweisungs-Cache 292 und einen Daten-Cache 294 für den Vektorprozessor 220 konfigurieren. Ein Ein-Chip-ROM 270, der Daten und Anweisungen für den allgemeinen Prozessor 210 und den Vektorprozessor 220 enthält, kann auch als ein Cache konfiguriert werden. Beim beispielhaften Ausführungsbeispiel enthält das ROM 270: Rücksetz- bzw. Reset- und Initialisierungs- bzw. Einleitungs-Prozeduren; Selbstüberprüfungs-Diagnoseprozeduren; Handhabungsmittel für einen Interrupt und eine Ausnahme; und Unterprogramme für eine Soundblaster-Emulierung; Subroutinen bzw. Unterprogramme für eine V. 34-Modem-Signalverarbeitung; allgemeine Telefonierfunktionen; 2-dimensionale und 3-dimensionale Graphik-Unterprogrammbibliotheken; und Unterprogramm-Bibliotheken für Ton- und Bild- bzw. Video-Standards, wie beispielsweise MPEG-1, MPEG-2, H.261, H.263, G.728 und G.723.

Fig. 3 verdeutlicht die Beziehungen zwischen Hardware und Software oder Firmware bei einer Anwendung der Multimediamkarte 100 in einem Host-Computersystem 300. Das Host-Computersystem 300 weist einen primären Prozessor 310 auf, der Programme ausführt, die in einem Hauptspeicher 320 abgespeichert sind. Bei dem beispielhaften Ausführungsbeispiel ist das Host-Computersystem 300 ein IBM-kompatibler Personalcomputer, der einen Mikroprozessor vom Typ x86 einschließt, und die Programme, die durch das Host-Computersystem 300 ausgeführt werden, umfassen ein Anwendungsprogramm 330, das unter einem Betriebssystem, wie beispielsweise Windows™ 95 oder NT läuft. Das Anwendungsprogramm 330 kann mit der Multimediamkarte 100 über Einheiten- bzw. Einrichtungstreiber 342 kommunizieren. Die Einrichtungstreiber 342 sind dem Einrichtungstreiber API des Betriebssystems angepaßt.

Die Einrichtungstreiber werden typischerweise mit jeder Multimediamkarte 100 bereitgestellt, da verschiedene Ausführungsbeispiele von Multimediamkarten 100 verschiedene Hardwareimplementierungen aufweisen können, wie beispielsweise sich unterscheidende Registerabbildungen und Interruptlevels bzw. Unterbrechungsprioritä-

ten. Die Einrichtungstreiber müssen die Steuersignale, die beim bestimmten Ausführungsbeispiel der Multimediamikarte 100 benötigt werden, geeignet in die Steuersignale transformieren bzw. umwandeln, wie dies durch den Einrichtungstreiber API des Betriebssystems bestimmt wird. Typischerweise wird das Betriebssystem einen für jede Einrichtung verschiedenen Einrichtungstreiber, wie beispielsweise einem Modemtreiber, einem Graphiktreiber und einen Tontreiber erwarten. So sind typischerweise drei getrennte Einrichtungstreiber für das Betriebssystem erforderlich, falls ein Ausführungsbeispiel der Multimediamikarte 100 die Funktionalität einer Tonkarte bzw. Audiokarte, eines Modems und einer Graphikkarte kombiniert.

Der Universal-Prozessor 210 im Multimedia-Prozessor 110 führt ein Echtzeit-Betriebssystem 360 aus, das Kommunikationen bzw. Nachrichtenübermittlungen mit Einrichtungstreibern 342 steuert. Der Universal-Prozessoren 21 führt auch allgemeine Aufgaben 370 aus. Der Vektorprozessor 220 führt Vektoraufgaben 380 aus.

Das Cache-Untersystem 230 (Fig. 2) koppelt auch allgemeine Prozessoren 210 und einen Vektorprozessor 220 mit zwei Systembussen: IOBUS 240 und FBUS 250. IOBUS 240 wird typischerweise mit einer geringeren Frequenz als der FBUS 250 betrieben. Einrichtungen mit einer geringeren Geschwindigkeit werden an den IOBUS 240 gekoppelt, während Einrichtungen mit einer höheren Geschwindigkeit mit dem FBUS 250 gekoppelt werden. Durch Trennen der Einrichtungen mit geringerer Geschwindigkeit von den Einrichtungen mit höherer Geschwindigkeit, werden die Einrichtungen mit geringer Geschwindigkeit gehindert, übermäßig auf die Leistungsfähigkeit der Einrichtungen mit höherer Geschwindigkeit einzuwirken.

Das Cache-Untersystem 230 dient auch als eine Schalttafel bzw. eine Schaltplatte für eine Kommunikation zwischen dem IOBUS 240, dem FBUS 250, dem allgemeinen Prozessor 210 und dem Vektorprozessor 220. Bei den meisten Ausführungsbeispielen vom Cache-Intersystem 230 sind mehrere gleichzeitige Zugriffe zwischen den Bussen und Prozessoren möglich. Z. B. ist der Vektorprozessor 220 in der Lage, mit dem FBUS 250 zur gleichen Zeit zu kommunizieren, zu der der Universal-Prozessor 210 mit dem IOBUS 240 kommuniziert. Bei einem Ausführungsbeispiel der Erfindung wird die Kombination der Schaltplatte und der Cachefunktion durch das Verwenden von direkten Abbildungstechniken für FBUS 250 und IOBUS 240 erzielt. Insbesondere kann auf die Einrichtungen am FBUS 250 und am IOBUS 240 durch den Universal-Prozessor 210 und den Vektorprozessor 220 mittels Standard-Speicherlesevorgängen und Standard-Speicherbeschreibvorgängen bei geeigneten Adressen zugegriffen werden.

Fig. 5 stellt die Speicherabbildung von einem Ausführungsbeispiel dar. Ein Speicherblock 510, d. h. der Adressraum von der Byteadresse Null zur Adresse 4M-1 wird durch das ROM 270 belegt (Die Einheiten M und G, die in dieser Beschreibung als Einheiten für Speicheradressen verwendet werden, stehen für die Zahlen 1.048.576 (d. h. 1024•1024) und 1.073.741.824 (d. h. 1024•1024•1024)). Ein Speicherblock 520, d. h. der Adressraum von einer Byte-Adresse 4M bis 8M-1 wird durch einen SRAM-Block 260 belegt. Ein Speicherblock 530, d. h. der Adressraum von einer Byte-Adresse 8M zu einer Adresse 72M-1 wird durch einen lokalen Speicher 120 belegt. Die Einrichtungen am FBUS 250 werden auf einen Speicherblock 540 abgebildet, der nach einer Byte-Adresse 72M beginnt und sich zu einer Byte-Adresse 77M erstreckt. Ein Speicherblock 550 wird für eine zukünftige Erweiterung reserviert. Die Einrichtungen am IOBUS 240 werden auf einen Speicherblock 560 abgebildet, der nach einer Byte-Adresse 125M beginnt und sich zu einer Byte-Adresse 128 M-1 erstreckt. Ein Speicherblock 570 wird auch für eine zukünftige Erweiterung reserviert. Ein Speicherblock 580, d. h. der Adressraum von einer Byte-Adresse 2G bis zu einer Adresse 4G-1 wird durch andere Host-Computereinrichtungen belegt und ein Zugriff erfolgt typischerweise über eine lokale Busschnittstelle bzw. Schnittstelle 255 für den lokalen Bus.

Der FBUS 250 (Fig. 2) steht mit einer Speichersteuereinrichtung 258, einer DMA-Steuereinrichtung 257, einer Schnittstelle 255 für einen lokalen Bus und einer Einrichtungsschnittstelle 252 in Verbindung, die entsprechend Schnittstellen für einen lokalen Speicher 120, einen lokalen Bus 105 und Wandler 132, 134, 142, 144, 146, 148 und 150 bereitstellt, wie dies in Fig. 1 dargestellt ist.

Die Speichersteuereinrichtung 258 steuert Lesevorgänge und Schreibvorgänge am lokalen Speicher 120. Beim beispielhaften Ausführungsbeispiel steuert die Speichersteuereinrichtung 258 eine Bank synchroner DRAMs (zwei 1M x 16-SDRAM-Chips), die konfiguriert sind, 24 bis 26 Adressbit und 32 Datenbit zu verwenden und die Merkmale aufweisen: (i) ein "CAS-vor-RAS"-Auffrischungsprotokoll, das mit einer programmierbaren Auffrischrate durchgeführt werden kann, (ii) teilweise Schreibvorgänge, die Lese-Modifizierte-Schreib-Operationen einleiten und (iii) eine interne Bankverschachtelung bzw. ein internes Bank-Interleaving. Die Speichersteuerung 258 bietet auch eine 1 : 1-Frequenzanpassung zwischen Speicher 120 und FBUS 250, eine manuelle "Vorladung beider Bänke" und Adress- und Daten-Warteschlangenvorgänge zum besseren Nutzen des FBUS 250. Bekannt ist, daß synchrone DRAMs bei solchen Frequenzen (80 MHz) effektiv betrieben werden und Standard-Fastpage- bzw. Standard-Schnells Seiten-DRAMs und erweiterte Datenausgabe-(EDO)-DRAMs auch verwendet werden können. DRAM-Steuereinrichtungen mit Fähigkeiten ähnlich zu einer Speichersteuereinrichtung 258 beim beispielhaften Ausführungsbeispiel sind in der Technik bekannt.

Die DRAM-Steuereinrichtung 257 steuert Direkt-Speicherzugriffe zwischen dem Hauptspeicher des Hostcomputers und dem lokalen Speicher eines Multimedia-Prozessors 200. Solche DMA-Steuereinrichtungen sind in der Technik bekannt. Bei einigen Ausführungsbeispielen ist eine Einrichtung für eine Speicherdatenbewegung eingeschlossen. Die Einrichtung zur Speicherdatenbewegung führt einen DMA (direkten Speicherzugriff) von einem Block des Speichers zu einem anderen Block des Speichers.

Die Schnittstelle 255 für den lokalen Bus implementiert das erforderliche Protokoll für Kommunikationen zwischen dem Hostcomputer über den lokalen Bus 105. Beim beispielhaften Ausführungsbeispiel bietet die Schnittstelle 255 für den lokalen Bus eine Schnittstelle zu einem 33-MHz-32-Bit-PCI-Bus. Solche Schnittstellen sind in der Technik bekannt.

Die Einrichtungsschnittstelle 252 bietet eine Hardwareschnittstelle für Einrichtungen wie beispielsweise Wandler 132, 134, 142, 144, 146, 148 und 150, die typischerweise auf einer Leiterplatte mit dem Multimedia-Prozessor 110 vorliegen. Die Einrichtungsschnittstelle 252 kann für die bestimmte Anwendung des Multimedia-Pro-

zessors 110 anwendungsspezifisch angepaßt werden. Insbesondere könnte die Einrichtungsschnittstelle 252 nur eine Schnittstelle für bestimmte Einrichtungen oder ICs bieten. Typische Einheiten innerhalb der Einrichtungsschnittstelle 252 bieten eine Schnittstelle für eine Verbindung zu Standard-ADCs, DACs oder CODECs. Entwürfe bzw. Aufbauformen für ADC-, DAC- und CODEC-Schnittstellen sind in der Technik bekannt und werden hier nicht weiter beschrieben. Andere Schnittstellen, die verwendet werden könnten, können eine ISDN-Schnittstelle für ein digitales Telefon und Schnittstellen für Busse, beispielsweise für einen Microchannel-Bus umfassen, sind darauf aber nicht beschränkt. Bei einem Ausführungsbeispiel des Multimedia-Prozessors 110 ist die Einrichtungsschnittstelle 252 ein ASIC, der programmiert werden kann, um eine gewünschte Funktionalität auszuführen bzw. zu bieten.

Der IOBUS 240 wird mit einer Frequenz (40 MHz) betrieben, die geringer als die Betriebsfrequenz (80 MHz) des Busses 250 ist. Mit dem IOBUS 240 sind ein Systemtaktgeber 242, ein UART (universelle asynchrone Empfänger-Sendereinheit) 243, ein Bitstrom-Prozessor 248 und eine Interrupt-Steuereinrichtung 245 gekoppelt. Der Systemtaktgeber 242 unterbricht den Prozessor 210 bei nominellen bzw. planmäßigen Zeitintervallen, die durch Schreiben in Register ausgewählt werden, die dem Systemzeitgeber 242 zugeordnet sind. Beim beispielhaften Ausführungsbeispiel ist der Systemzeitgeber 242 ein Standard-Intel-8254-kompatibler Intervalzeitgeber mit drei unabhängigen 16-Bit-Zählern und sechs programmierbaren Zählerbetriebsarten.

UART 243 ist eine serielle Schnittstelle, die zu den bekannten bzw. allgemein verwendeten integrierten 16450-UART-Schaltung kompatibel ist und dient für eine Anwendung bei Modem- oder Faxanwendungen, die einen seriellen Standard-Kommunikations-(“COM”)-Anschluß eines Personalcomputers erforderlich machen.

Der Bitstromprozessor 245 ist ein fester Hardwareprozessor, der bestimmte Funktionen an einem Eingabe- oder Ausgabe-Bitstrom ausführt. Beim beispielhaften Ausführungsbeispiel führt der Bitstrom-Prozessor 245 anfängliche oder abschließende Stufen einer MPEG-Codierung oder -Decodierung durch. Insbesondere führt der Bitstrom-Prozessor 245 eine Codierung und Decodierung mit variabler Länge durch (Huffman) und ein Packen bzw. Entpacken von Bilddaten in einem “zick-zack”-Format. Der Bitstrom-Prozessor 245 wird parallel zum Universal-Prozessor 210 und dem Vektorprozessor 220 betrieben und davon gesteuert. Die Prozessoren 210 und 220 konfigurieren den Bitstrom-Prozessor 245 über Steuerregister. Die ebenfalls anhängige Patentanmeldung mit dem Titel “Verfahren und Vorrichtung zum Verarbeiten von Videodaten”, die durch Bezug aufgenommen wird, beschreibt ein beispielhaftes Ausführungsbeispiel eines Bitstrom-Prozessors 245.

Die Interrupt-Steuereinrichtung 248 bzw. Unterbrechungssteuereinrichtung steuert Unterbrechungen bzw. Interrupts des Universal-Prozessors 210 und unterstützt viele Interruptprioritäten. Ein Maskenregister wird bereitgestellt, um zu ermöglichen, daß jede Interruptpriorität individuell maskiert wird. Beim beispielhaften Ausführungsbeispiel ist die interrupt-Steuereinrichtung 245 programmierbar und implementiert das Standard-Intel-8259-Interruptsystem, das bei x86-Personalcomputern üblich ist. Ein Interrupt mit höchster Priorität (Stufe 0) wird dem Systemzeitgeber 242 zugewiesen. Prioritätsstufen 1, 2, 3 und 7 werden entsprechend einem virtuellen Vollbild- bzw. Frame-Puffer, einer DMA-Steuereinrichtung 257 und einer Einrichtungsschnittstelle 252, einem Bitstrom-Prozessor 245, einer Schnittstelle 255 für den lokalen Bus und dem UART 243 zugewiesen. Interrupt-Prioritätsstufen 4, 5 und 6 sind bei dem beispielhaften Ausführungsbeispiel nicht zugewiesen. Der virtuelle Frame-Puffer, der bei einigen Ausführungsbeispielen eingeschlossen ist, emuliert bei der Prioritätsstufe 1 einen Standard-VGA-Frame-Puffer bzw. -Vollbildpuffer.

Fig. 4 stellt ein Blockdiagramm des Cache-Untersystems 230 dar. Ein SRAM-Block 260 ist in vier Speicherbänke aufgeteilt, um einen Anweisungscache 262 und einen Datencache 264 für eine Anwendung mit dem allgemeinen Prozessor 210 wie auch einen Anweisungscache 292 und einen Datencache 294 für eine Anwendung mit dem Vektorprozessor 220 auszubilden. Der SRAM-Block 260 enthält auch einen Identifizierungszeichen- bzw. Kennzeichenabschnitt 406, der für jede der Speicherbänke unterteilt ist. Der SRAM-Block 260 ist eine Speicherschaltung mit zwei Anschlüssen, mit einem Leseanschluß 440 und einem Schreibanschluß 430, so daß ein gleichzeitiges Lesen und Schreiben des SRAM-Blocks 260 unterstützt wird. Das Cache-Untersystem 230 enthält auch ein ROM-Cache 270 mit einem Kennzeichen-Feld 472. Wie vorstehend erläutert, enthält der ROM-Cachespeicher 270 häufig verwendete Anweisungen und Daten für den allgemeinen Prozessor 210 und den Vektorprozessor 220. Obwohl das Kennzeichen-Feld 472 nicht modifiziert werden kann, können einzelne Adressen als ungültig markiert werden, so daß diese Daten oder Anweisungen aus dem Speicher gebracht werden können, um anstelle der Daten oder Anweisungen im ROM 270 verwendet zu werden.

Eine Datenpipeline 410 führt die Daten-Schalttafel-Funktion des Cache-Untersystems 230 aus. Die Datenpipeline 410 kann mehrere simultane bzw. gleichzeitige Datenkommunikationswege zwischen dem IOBUS 240, dem FBUS 250, dem Universal-Prozessor 210, dem Vektorprozessor 220 und dem SRAM-Block 260 erzeugen. Ähnlich führt die Adresspipeline 420 Schalttaffunktionen für Adressen aus. Beim Ausführungsbeispiel der Fig. 4 verwenden IOBUS 240 und FBUS 250 einen Zeitmultiplexbetrieb für Adreß- und Datensignale. Die Cachesteuerung 280 stellt die Steuerleitungen zur Datenpipeline 410 und zur Adresspipeline 420 bereit, um die Kommunikationskanäle geeignet zu konfigurieren.

Bei irgendeinem Ausführungsbeispiel des Cache-Untersystems 230 wird ein Protokoll, das auf einem Vorgang bzw. einer Transaktion beruht, verwendet, um alle Lese- und Schreibvorgänge zu unterstützen. Irgendeine Einheit, die mit dem Cache-Untersystem 230 gekoppelt ist, wie der allgemeine Prozessor 210, der Vektorprozessor 220 oder die verschiedenen Einrichtungen an IOBUS 240 und FBUS 250 kann eine Aufforderung an das Cache-Untersystem 230 setzen. Eine solche Aufforderung wird durch einen Einrichtungs-Identifizierungscode (“Einrichtungs-ID”) und eine Adresse der aufgeforderten bzw. abgefragten Speicherstelle ausgebildet. Jede Einheit weist eine getrennte Einrichtungs-ID auf, und das Cache-Untersystem 230 kann die Aufforderungen bzw. Anfragen auf der Einrichtungs-ID der Einheit, die die Abfrage durchführt, beruhend hinsichtlich der Priorität einteilen. Wenn die Daten bei der abgefragten Adresse verfügbar werden, antwortet das Cache-Untersystem mit der Einrichtungs-ID, einem Transaktions- bzw. Vorgangs-Identifizierungscode (“Transaktions-ID”), der Adresse

und den abgefragten Daten. Falls die angeforderte bzw. abgefragte Adresse nicht im SRAM-Block 260 oder ROM 270 enthalten ist, ist das Cache-Untersystem 230 nicht über mehrere Taktzyklen in der Lage, auf die bestimmte Aufforderung zu antworten, während die Daten bei der Speicheradresse zurückbekommen werden. Während die Daten einer ersten Aufforderung zurückbekommen werden, ist das Cache-Untersystem 230 jedoch in der Lage, eine zweite Aufforderung von einer dazu verschiedenen Einheit mit einer verschiedenen Einrichtungs-ID zu verarbeiten. Auf diese Art und Weise blockiert eine anhängige Aufforderung nachfolgende Aufforderungen von anderen Einheiten nicht. Ferner kann das Cache-Untersystem 230 eine Leseaufforderung und eine Schreibaufforderung gleichzeitig in einem einzelnen Zyklus handhaben.

Wie vorstehend erläutert ist der SRAM-Block 260 in vier Speicherbänke aufgeteilt. Der SRAM-Block 260 weist zwei Eingänge bzw. Anschlüsse auf, d. h. er weist einen Leseanschluß 440 und einen Schreibanschluß 430 auf, so daß der SRAM-Block 260 bei irgendeinem Zyklus eine Leseaufforderung und eine Schreibaufforderung annehmen kann. Der Kennzeichen-Abschnitt 406 des SRAM-Blocks 260 muß zwei Leseanschlüsse aufweisen, um die gleichzeitigen Lese- und Schreibaufforderungen zu unterstützen. Daher können die Adresse, die beim Leseanschluß 440 verwendet wird, wie auch die Adresse, die beim Schreibanschluß 430 verwendet wird, mit internen Cache-Kennzeichen für Treffer- oder Fehlbedingungen gleichzeitig verglichen werden. Der Kennzeichen-Abschnitt 406 enthält auch einen getrennten Schreibanschluß, so daß die geeigneten Kennzeichenfelder auch geändert werden können, während die Schreibaufforderung am Schreibanschluß 430 durchgeführt wird.

Abhängig von den Beschränkungen des Gesamtsystems kann das Cache-Untersystem 230 mit entweder Schreib-Zurück- oder Schreib-Durch-Cacheprinzipien bzw. -grundsätzen verwendet werden. Ferner kann die Cache-Zeilengröße bei einigen Ausführungsbeispielen zum weiteren Erhöhen der Geschwindigkeit zweimal so groß wie die Datenbreite gemacht werden. Bei diesen Ausführungsbeispielen muß für "Systemverwaltungs"-Zwecke jede Cachezeile zwei gültige Bit und zwei Speichermarken bzw. Dirtybit zugewiesen werden, da jede Cachezeile zwei Vektoren enthält. Der SRAM-Block 260 sollte also global alle gültigen Bit löschen, falls ein globales Löschsignal empfangen wird. Bei anderen Ausführungsbeispielen werden individuelle bzw. einzelne Löschsignale für jede Bank im SRAM-Block 260 unterstützt.

Fig. 6 ist ein Blockdiagramm eines Ausführungsbeispiels einer Datenpipeline 410. Da das Cache-Untersystem 230 sowohl ein Cachesystem als auch eine Schalttafel für den IOBUS 240, den FBUS 250, den Universal-Prozessor 210 und den Vektorprozessor 220 ist, sollten die Busse und der Prozessor in der Lage sein, entweder durch den Cache oder direkt zu kommunizieren, falls der Cachespeicher durch eine andere Einrichtung verwendet wird. Die Prozessoren sind im allgemeinen schneller als die Einrichtungen an den Bussen; daher werden die Prozessoren im allgemeinen den Cache bei Schreibvorgängen verwenden und dem Cache-Rückschreibsystem ermöglichen, die Daten auf die geeignete Buseinrichtung zu setzen. Ähnlich fordern die Prozessoren allgemeine Aufforderungsinformationen eher vom Cache als direkt von den Einrichtungen. Falls der Cachespeicher die aufgeforderten Daten nicht enthält, verlassen sich die Prozessoren typischerweise auf das Cache-Untersystem, um die angeforderten Daten in den Cachespeicher auszulesen und die Daten für die Prozessoren zu erzeugen. Wenn der Cachespeicher jedoch besetzt ist bzw. tätig ist, können die Prozessoren direkt auf die Busse zugreifen.

Daten werden vom Universal-Prozessor 210 über einen IOMUX 630 zum IOBUS 240 übertragen. Daten vom IOBUS 240 zum Universal-Prozessor 210 laufen durch einen GP Lese MUX 620. Von entweder dem SRAM-Block 260 oder vom ROM 207 werden Daten zum Universal-Prozessor 210 über einen Cache-Lese-MUX 650 und einen GP-Lese-MUX 620 übertragen. Vom Universal-Prozessor 210 werden Daten zum SRAM-Block 260 über einen Cache-Schreib-MUX 610 übertragen. Der Cache-Lese-MUX 650, der Cache-Schreib-MUX 610, der IO-MUX 630 und der GP-Lese-MUX 620 können konventionelle Multiplexer sein und können interne Latche bzw. Zwischenspeicher oder Register enthalten, sofern dieses für Zeitbeschränkungen erforderlich ist. Die Ausfall-Steuerleitungen (nicht dargestellt) der Multiplexer werden durch die Cache-Steuereinrichtung 280 (Fig. 4) vorgeschrieben. Vom Universal-Prozessor 210 werden Daten zum FBUS 250 über den Cache-Schreib-MUX 610 und den FBUS-MUX 640 übertragen. Vom FBUS 250 werden Daten zum Universal-Prozessor 210 kanalisiert über einen Puffer 660, einen Cache-Lese-MUX 650 und den GP-Lese-MUX 620 geführt. Um diese Funktionen auszuführen, kann der Puffer 660 ein konventioneller Puffer, ein Latch oder ein Register sein.

Der Universal-Prozessor 210 kann den Vektorprozessor 220 über Steuerleitungen 212 (Fig. 2) steuern. Eine direkte Datenübertragung zwischen dem Universal-Prozessor 210 und dem Vektorprozessor 220 ist im allgemeinen nicht erforderlich, kann aber über den SRAM-Block 260 oder irgendeine andere der Einrichtungen ermöglicht werden, da die beiden Prozessoren sich eine gemeinsame Speicherabbildung teilen.

Daten vom ROM 270 und SRAM-Block 260 wandern zum IOBUS 240 über den Cache-Lese-MUX 650 und den IO-MUX 630. Daten vom IOBUS 240 zum SRAM-Block 260 werden über den Cache-Schreib-MUX 610 geführt. Daten vom IOBUS 240 zum FBUS 250 werden durch den Cache-Schreib-MUX 610 und den FBUS-MUX 640 geführt. Daten vom FBUS 250 für den IOBUS 240 werden durch den Zwischenspeicher bzw. Puffer 660, den Cache-Lese-MUX 650 und den IO-MUX 630 geführt. Daten für den IOBUS 240 vom Vektorprozessor 220 werden durch den Cache-Schreib-MUX 610 und den IO-MUX 630 geführt. Daten vom IOBUS 240 zum Vektorprozessor 220 laufen durch den Cache-Lese-MUX 650. Bei einigen Ausführungsbeispielen ist die direkte Pipeline bzw. Richtungspipeline für Daten vom Vektorprozessor 220 zum IOBUS 240 zum Vereinfachen des Aufbaus der Datenpipeline 410 weggelassen. Da die Bandbreite des Vektorprozessors 220 viel größer als die Bandbreite des IOBUS 240 ist, wird ein direkter Kommunikationsweg vom Vektorprozessor 220 zum IOBUS 240 hinsichtlich der Verarbeitungszeit des Vektorprozessors 220 sehr ineffektiv.

Daten vom SRAM-Block 260 und ROM 270 für den FBUS 250 laufen durch den Cache-Lese-MUX 650 und den FBUS-MUX 640. Daten vom FBUS 250 zum SRAM-Block 260 laufen durch den Puffer 660 und den Cache-Schreib-MUX 610. Daten vom FBUS 250 können den Vektorprozessor 220 über den Puffer 660 und den Cache-Lese-MUX 650 direkt erreichen. Daten für den FBUS 250 können vom Vektorprozessor 220 auch direkt über den Cache-Schreib-MUX 610 und den FBUS-MUX 640 kommen.

Daten vom Vektorprozessor 220 wandern zum SRAM-Block 260 über den Cache-Schreib-MUX 610. Daten vom SRAM-Block 260 und vom ROM 270 werden durch den Cache-Lese-MUX 650 zum Vektorprozessor 220 geführt.

Fig. 7 ist ein detailliertes Blockdiagramm eines zweiten Ausführungsbeispiels der Datenpipeline 410. Da die Funktionalität des Ausführungsbeispiels der Fig. 7 ähnlich der Funktionalität des Ausführungsbeispiels der Fig. 6 ist, werden nur die Unterschiede zwischen den Ausführungsbeispielen in Einzelheiten erörtert. Die allgemeine Organisation der Elemente bei jedem Ausführungsbeispiel wird jedoch auch beschrieben. In Fig. 7 ist der Cache-Lese-MUX 650 gegen den Cache-Lese-MUX 750 und einen MUX-Latch 751 bzw. MUX-Zwischen-speicher ausgetauscht. Der Puffer 660 ist gegen einen Lese-Latch 760 ausgetauscht. Der FBUS-MUX 640 ist durch einen FBUS-MUX 740, ein Rückschreib-(WB)-Datenlatch 741, einen Speicherschreib-Latch 742 und einen Speicherschreib-Latch 743 ersetzt. Die Latche bzw. Zwischenspeicher beim Ausführungsbeispiel der Fig. 7 werden zur Pipelineverarbeitung der Datenpipeline verwendet. Der Cache-Schreib-MUX 610 wird gegen einen Cache-Schreib-MUX 710, einen Schreib-Daten-Latch 712, eine Einrichtung zum Ausrichten 713 und einen IO- bzw. EIN/AUS-Schreib-Latch 711 ausgetauscht. Der IO-MUX 630 wird durch einen IO- bzw. EIN/AUS-Lese-Latch 731 und einen IO- bzw. EIN/AUS-Lese-Latch 732 ausgetauscht. Der GP-Lese-MUX 620 wird durch einen IO- bzw. EIN/AUS-Schreib-Latch 721 und einen Microcache 722 ersetzt.

Der Microcache 722 koppelt den Hauptcache, den SRAM-Block 270 und das ROM 260 mit dem Universal-Prozessor 210. Der Microcache 722 ist in einen Micro-Anweisungscache und einen Micro-Daten-Cache aufgeteilt, von denen jeder ein Kennzeichen-Teil 822 (Fig. 8), Etikett- bzw. Kennzeichen-Vergleicher und gültige Bit umfassen. Der Microcache 722 arbeitet als ein Vorabrufpuffer. Die Adresse einer Aufforderung vom Universal-Prozessor 210 wird zuerst mit dem Kennzeichen-Teil 822 des Microcaches 722 verglichen. Falls beim Microcache ein Fehlzustand auftritt (d. h., keine Übereinstimmung innerhalb des Microcache-Kennzeichens 822 vorliegt), wird die Adresse der Aufforderung mit der Adresse und einer anderen Steuerinformation zum Hauptcache gesendet. Um den Microcache 722 zu vereinfachen, müssen Übereinstimmungen eines Kennzeichens bei Datenschreibvorgängen vom Universal-Prozessor im Microcache 722 die Microcacheadresse ungültig machen, so daß die geschriebenen Daten zum Hauptcache gesendet werden müssen. Auf diese Art und Weise kann ein Cachezusammenhang ohne komplexe Rückschreib- oder Schreibvorgänge durch Aufbauformen am Microcache 722 beibehalten werden.

Fig. 8 stellt ein Blockdiagramm eines Ausführungsbeispiels der Adresspipeline 420 dar. Die FBUS-Schnittstelle 850 besteht aus einer Adressenwarteschlange für vier Einträgen und einem Rückschreiblatch. Die FBUS-Schnittstelle 850 kann gleichzeitig ein anhängiges Lesen vom Anweisungscache 662, ein anhängiges Lesen vom Anweisungscache 292, eine Schreibaufforderung vom Datencache 294 und eine Rückschreibaufforderung vom Datencache 294 unterstützen. Die Adressen für Schreibaufforderungen werden zum Schreib-Adress-MUX 810 gesendet, während die Adressen für Leseaufforderungen zum Lese-Adress-MUX 820 gesendet werden. Die Cachesteuereinrichtung 280 (Fig. 2) führt eine Arbitration bzw. Entscheidung zwischen Aufforderungen vom Universal-Prozessor 210, vom Vektorprozessor 220, vom IOBUS 240 und vom FBUS 250 aufgrund der Einrichtungs-ID der Aufforderung durch. Die Cachesteuerung 280 konfiguriert dann die verschiedenen Multiplexer der Datenpipeline 410 und der Datenpipeline 420, um die Aufforderungen handzuhaben. Ein Arbitrationsschema bzw. Entscheidungsschema kann aufgrund einer Bestimmung oder Abschätzung der Wichtigkeit jeder Einrichtung entschieden werden. Typischerweise wird dem Universal-Prozessor 210 die höchste Priorität gegeben. Wie vorstehend erläutert, ist das Cache-Untersystem 230 in der Lage, Lese- und Schreibvorgänge gleichzeitig durchzuführen. Daher sind getrennte Vergleicher für Lese- und Schreibaufforderungen erforderlich. Der Vergleicher 811 wird zum Vergleichen der Schreibadresse vom Schreib-Adress-MUX 810 mit den Adressen verwendet, die über einen Schreib-Kennzeichen-Anschluß 406-1 empfangen werden, um zu bestimmen, ob sich die Schreibadresse der momentanen Aufforderung im Cache befindet. Falls sich die Adresse im Cache befindet, wird der Cachespeicher beim Zusammenpassen bzw. Abstimmen der Cachespeicherstelle mit den neuen Daten aktualisiert. Falls sich die Adresse nicht im Cache befindet, werden die Adresse und die Daten bei einer nicht verwendeten Cachestelle oder der Stelle in den Cachespeicher geschrieben, auf die unlängst zuletzt zugegriffen wurde. Schließlich werden die Daten zu der geeigneten, direkt abgebildeten Einrichtung unter Verwendung eines Rückschreibens oder Schreibens durch Cachetechniken gesendet.

Der Vergleicher 821 wird zum Vergleichen der Leseadresse der Leseaufforderungen vom Lese-Adress-MUX 820 und der Adressen verwendet, die durch den Lese-Kennzeichen-Anschluß 406-2 empfangen werden. Falls ein Kennzeichen auf die Leseadressen abgestimmt ist bzw. mit denen übereinstimmt, werden die Daten, die zu dem Kennzeichen gehören, unter Verwendung der Datenpipeline 410 zur auffordernden Einrichtung gesendet. Wie vorstehend erläutert, werden die Daten mit einer Einrichtungs-ID, einer Transaktions-ID und der angeforderten Adresse zurückgegeben, falls das Transaktionsprotokoll verwendet wird. Falls keine Kennzeichen mit der Leseadresse übereinstimmen, muß das Cache-Untersystem 230 die angeforderten Daten von der geeigneten Direktspeicherabbildungseinrichtung bzw. im Direktspeicher abgebildeten Einrichtung wiedergewinnen bzw. auslesen. Wenn die angeforderten Daten ausgelesen werden, werden die angeforderten Daten, die Einrichtungs-ID, die Transaktions-ID und die Adresse zur anfordernden Einrichtung gesendet. Während die Daten für eine erste Anforderung abgerufen werden, ist das Cache-Untersystem 230 in der Lage, einer zweiten Leseaufforderung zu dienen, so daß eine zweite Einrichtung, die den Cache benötigt, nicht durch die erste Einrichtung blockiert wird.

Bei den verschiedenen Ausführungsbeispielen des Aufbaus können auch andere Implementierungen von Datenpipelines, Schalttafeln, Adresspipelines, Cache-Untersystemen, Multiplexern, Zwischenspeichern, Bussen, Prozessoren etc. Anwendung finden und alternative Merkmale können zum Erzeugen eines Digitalsignalprozessors verwendet werden.

## Querverweise

Dieses Patentdokument ist auf die nachfolgenden gleichzeitig angemeldeten Patentanmeldungen in deren Gesamtheit bezogen und umfaßt diese:

5 U.S. Pat-Anm.-Nr. UNKNOWN1, Anwaltsdokument Nr. M-4355, mit dem Titel "Single-Instruction-Multiple-  
Data Processing in a Multimedia Signal Processor";  
U.S. Pat-Anm.-Nr. UNKNOWN2, Anwaltsdokument Nr. M-4365, mit dem Titel "Efficient Context Saving and  
Restoring in Multiprocessors"  
10 U.S. Pat-Anm.-Nr. UNKNOWN3, Anwaltsdokument Nr. M-4366, mit dem Titel "System and Method for  
Handling Software Interrupts with Argument Passing";  
U.S. Pat-Anm.-Nr. UNKNOWN4, Anwaltsdokument Nr. M-4367, mit dem Titel "System and Method for  
Handling Interrupts and Exception Events in an Asymmetric Multiprocessor Architecture",  
U.S. Pat-Anm.-Nr. UNKNOWN5, Anwaltsdokument Nr. M-4368, mit dem Titel "Methods and Apparatus for  
15 Processing Video Data";  
U.S. Pat-Anm.-Nr. UNKNOWN6, Anwaltsdokument Nr. M-4369, mit dem Titel "Single-Instruction-Multiple-  
Data Processing Using Multiple Banks of Vector Registers"; und  
U.S. Pat-Anm.-Nr. UNKNOWN7, Anwaltsdokument Nr. M-4370, mit dem Titel "Single-Instruction-Multiple-  
Data Processing with Combined Scalar/Vector Operations".

## 20 Patentansprüche

1. Integrierter Digitalsignalprozessor, aufweisend:  
einen Universal-Prozessor (210); und  
25 einen Vektorprozessor (220), der parallel zum Universal-Prozessor betreibbar ist
2. Integrierter Digitalsignalprozessor nach Anspruch 1, dadurch gekennzeichnet, daß der Universal-Prozessor (210) aufweist:  
einen Satz Skalarregister;  
eine Anweisungs-Decodiereinheit und  
30 einen Verarbeitungskern (200) der eine Vielzahl skalarer Werte entsprechend Anweisungen manipuliert, die mittels der Anweisungs-Decodiereinheit decodiert werden
3. Integrierter Digitalsignalprozessor nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß der Vektorprozessor (220) aufweist:  
einen Satz Vektorregister;  
35 eine zweite Anweisungs-Decodiereinheit und  
einen zweiten Verarbeitungskern, der eine Vielzahl von Vektorwerten entsprechend Anweisungen manipuliert, die durch die zweite Anweisungs-Decodiereinheit decodiert werden
4. Integrierter Digitalsignalprozessor nach einem der vorhergehenden Ansprüche, ferner gekennzeichnet durch:  
ein Cache-Untersystem (230), das mit dem Universal-Prozessor (210) und dem Vektorprozessor (220) gekoppelt ist, wobei das Cache-Untersystem (230) einen Speichercache aufweist
5. Integrierter Digitalsignalprozessor nach Anspruch 4, dadurch gekennzeichnet, daß das Cache-Untersystem (230) aufweist:  
einen Cache-Leseanschluß; und  
45 einen Cache-Schreibanschluß;  
wobei das Cache-Untersystem gleichzeitige Zugriffe auf den Cache-Leseanschluß und den Cache-Schreibanschluß unterstützt
6. Integrierter Digitalsignalprozessor nach Anspruch 4 oder 5, dadurch gekennzeichnet, daß das Cache-Untersystem (230) ferner aufweist:  
eine Datenpipeline (410), die mit dem Universal-Prozessor (210) und dem Vektorprozessor (220) gekoppelt ist; und  
50 eine Adresspipeline (420), die mit dem Universal-Prozessor (210) und dem Vektorprozessor (220) gekoppelt ist; und  
wobei der Speichercache aufweist:  
einen SRAM-Cache, der mit der Datenpipeline und der Adresspipeline gekoppelt ist; und  
einen ROM-Cache, der mit der Datenpipeline und der Adresspipeline gekoppelt ist
7. Integrierter Digitalsignalprozessor nach einem der Ansprüche 4 bis 6, ferner gekennzeichnet durch:  
einen ersten Bus, der mit dem Cache-Untersystem (230) gekoppelt ist; und  
60 einen zweiten Bus, der mit dem Cache-Untersystem gekoppelt ist
8. Integrierter Digitalsignalprozessor nach Anspruch 7, dadurch gekennzeichnet, daß der erste Bus eine erste Bus-Bandbreite aufweist, die größer als eine zweite Bus-Bandbreite des zweiten Busse ist
9. Integrierter Digitalsignalprozessor nach Anspruch 7 oder 8, ferner gekennzeichnet durch:  
einen Bitstrom-Prozessor, der mit dem zweiten Bus gekoppelt ist; und  
65 eine Schnittstelle (255) für einen lokalen Bus, die mit dem ersten Bus gekoppelt ist
10. Integrierter Digitalsignalprozessor nach Anspruch 9, dadurch gekennzeichnet, daß die Schnittstelle (255) für den lokalen Bus mit einem lokalen Bus eines primären Prozessors eines Computersystems gekoppelt ist
11. Prozessor nach Anspruch 9 oder 10, dadurch gekennzeichnet, daß das Cache-Untersystem (230) zum Bereitstellen einer Vielzahl von Kommunikationsbahnen zwischen dem Vektorprozessor (220), dem Uni-

versal-Prozessor (210), dem ersten Bus und dem zweiten Bus konfigurierbar ist.  
12. Integrierter Digitalsignalprozessor nach einem der Ansprüche 7 bis 11, dadurch gekennzeichnet, daß das Cache-Untersystem (230) über eine erste Cacheaufforderung zugreifbar ist.  
13. Integrierter Digitalsignalprozessor nach Anspruch 12, dadurch gekennzeichnet, daß das Cache-Untersystem (230) zum Annehmen einer zweiten Cache-Aufforderung vor dem Abschließen der ersten Cache-Aufforderung in der Lage ist, falls zum Abschließen der ersten Cache-Aufforderung eine Vielzahl von Zyklen erforderlich ist.

5

Hierzu 8 Seite(n) Zeichnungen

10

15

20

25

30

35

40

45

50

55

60

65

**- Leerseite -**

**THIS PAGE BLANK (USPTO)**

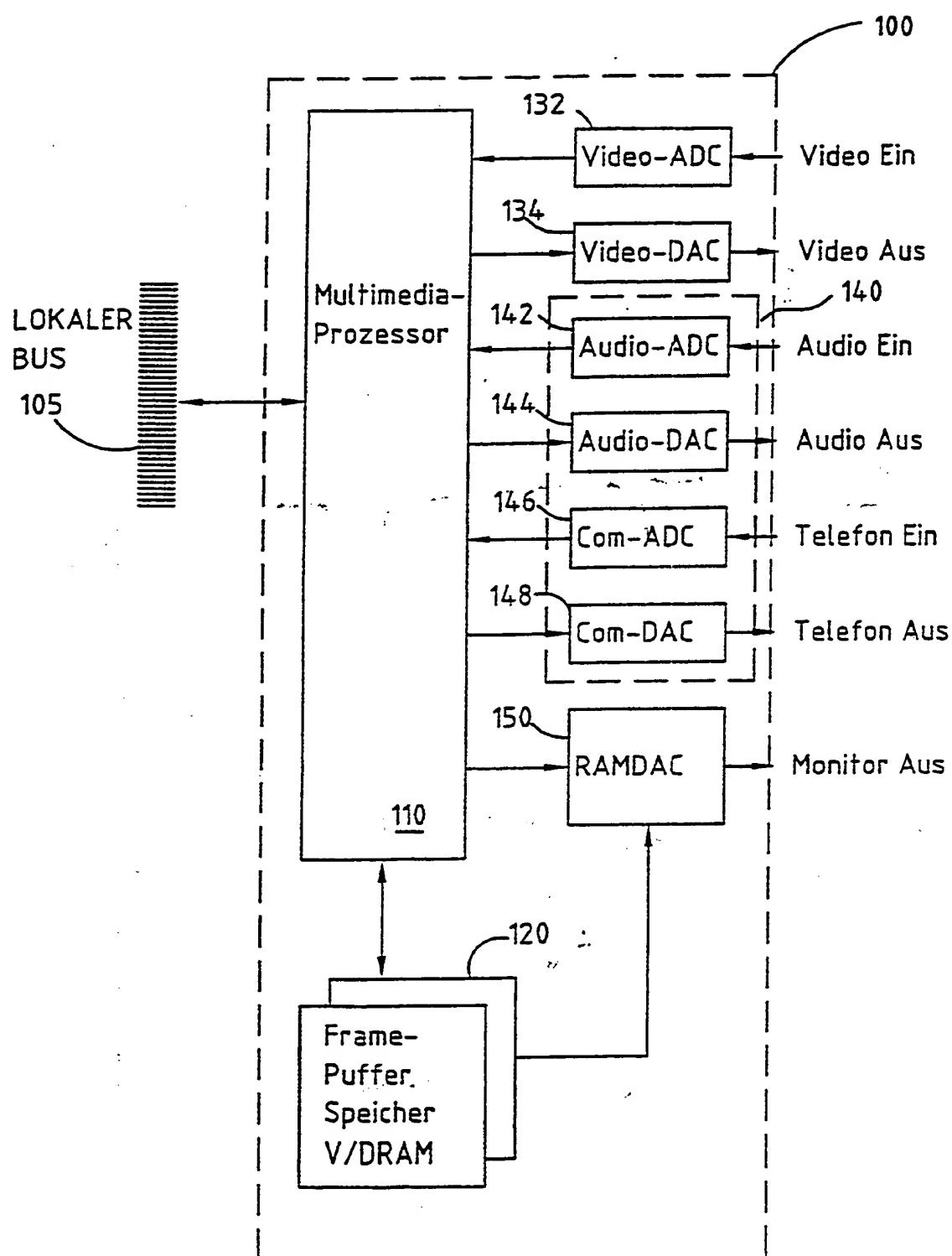


Fig. 1

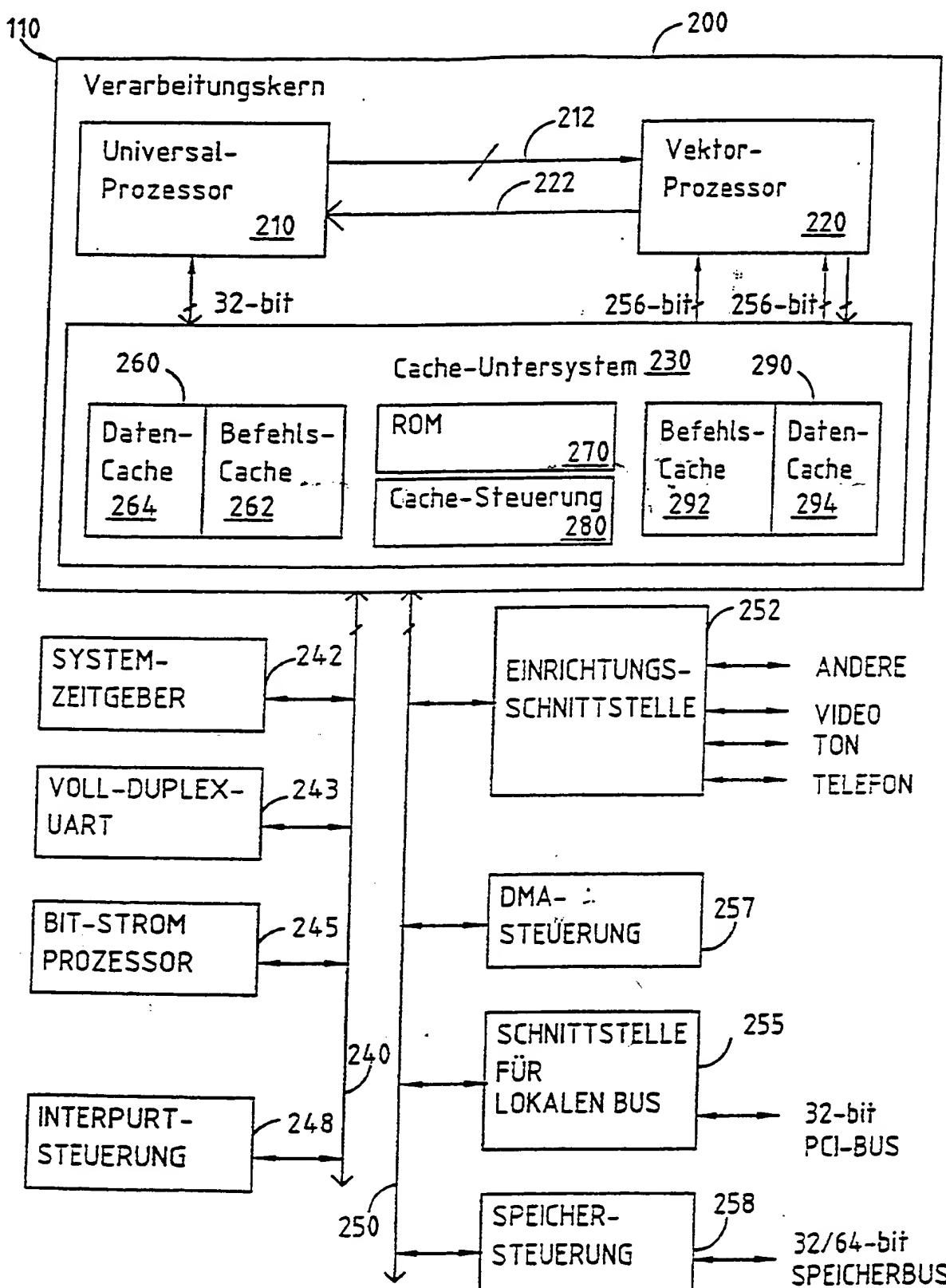


Fig. 2

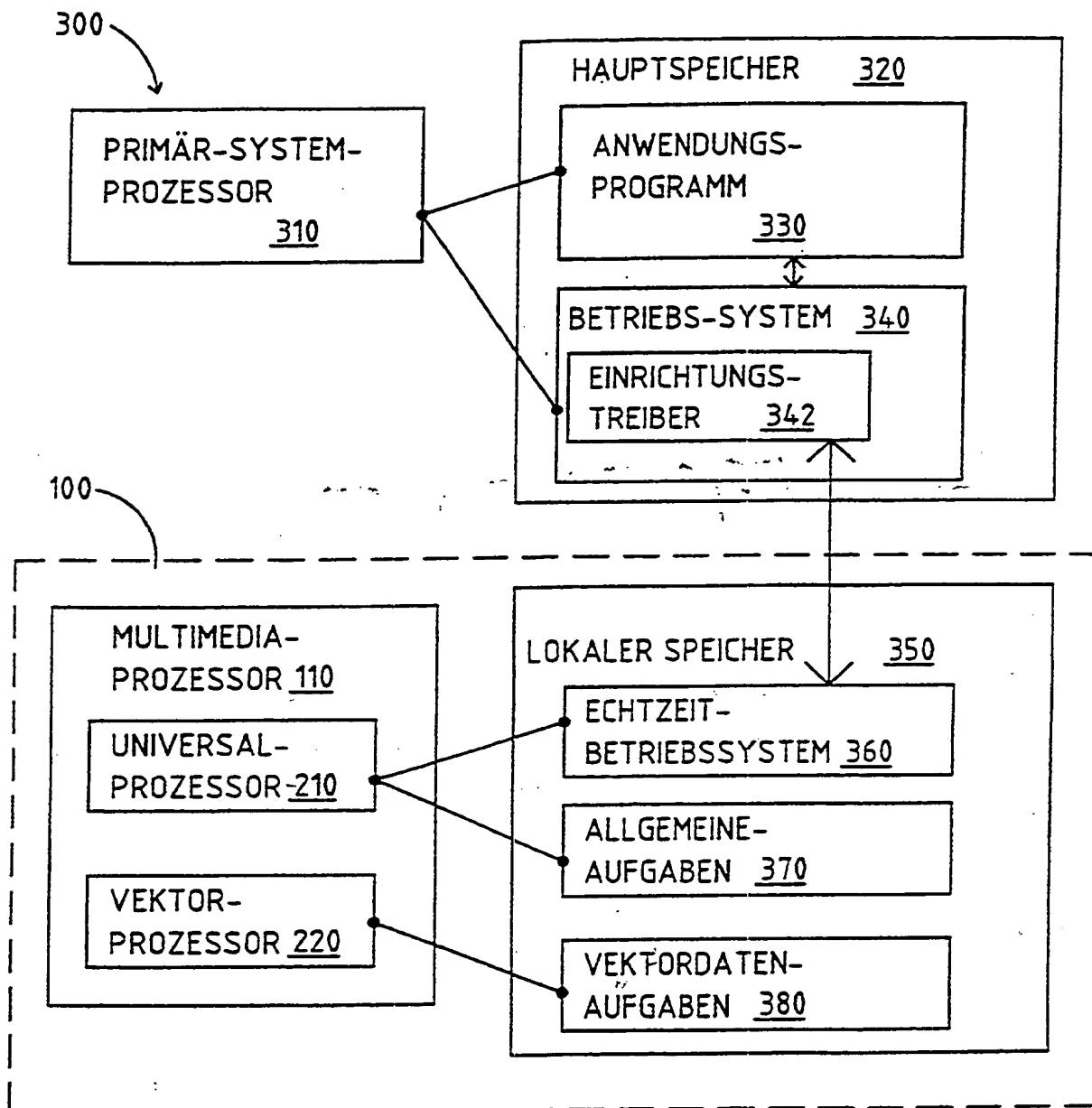


Fig. 3

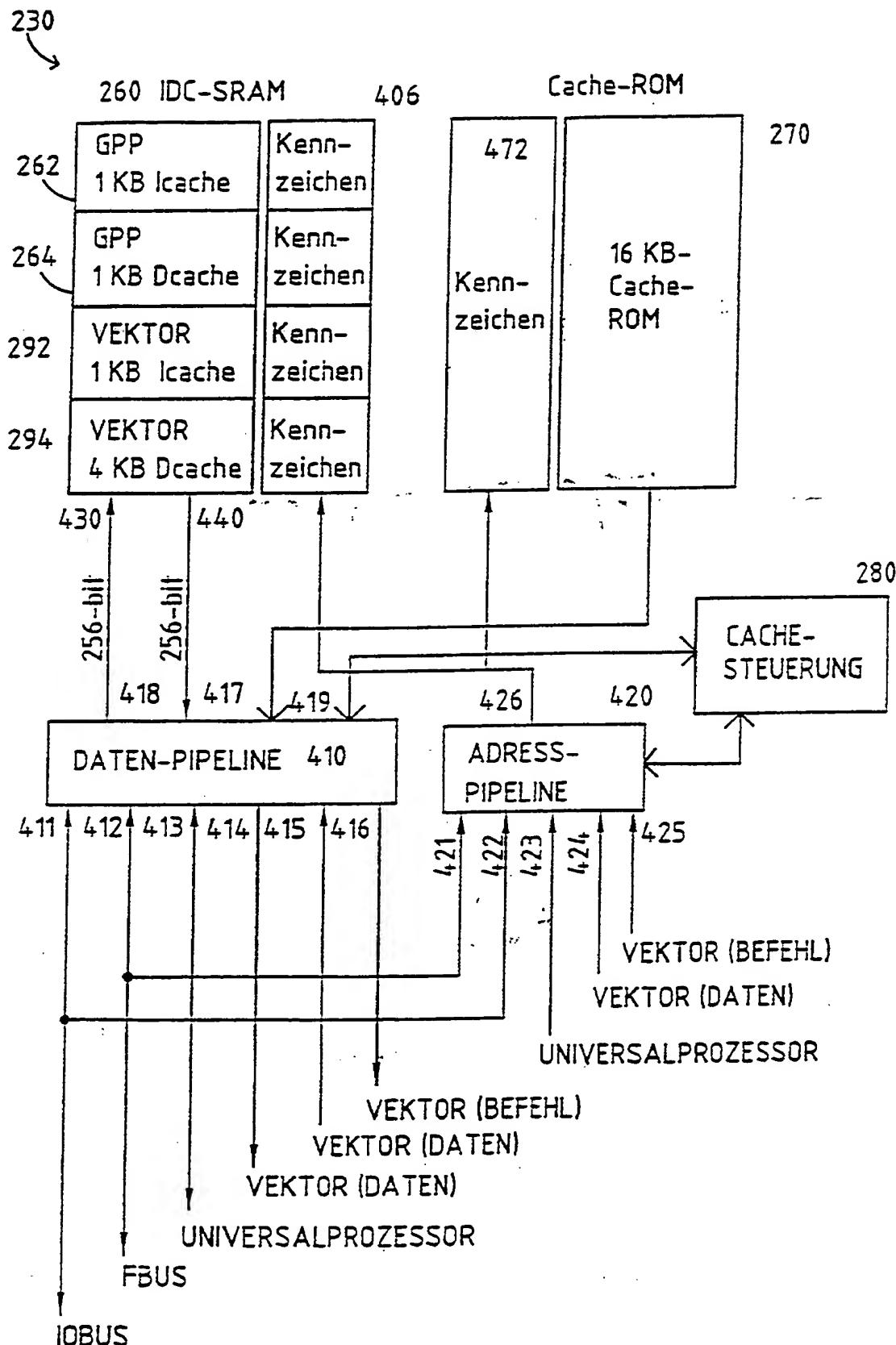


Fig. 4

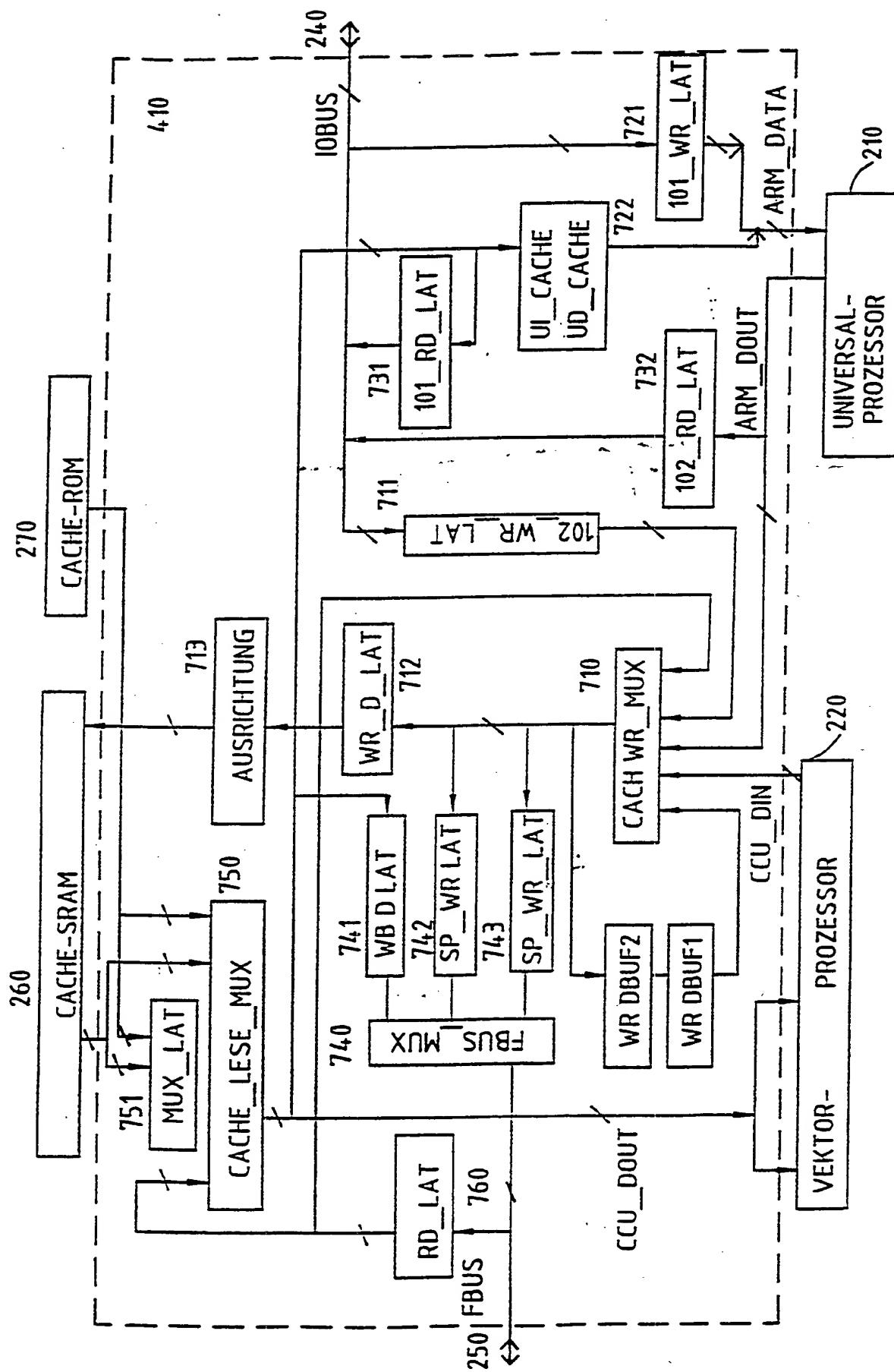


Fig. 7

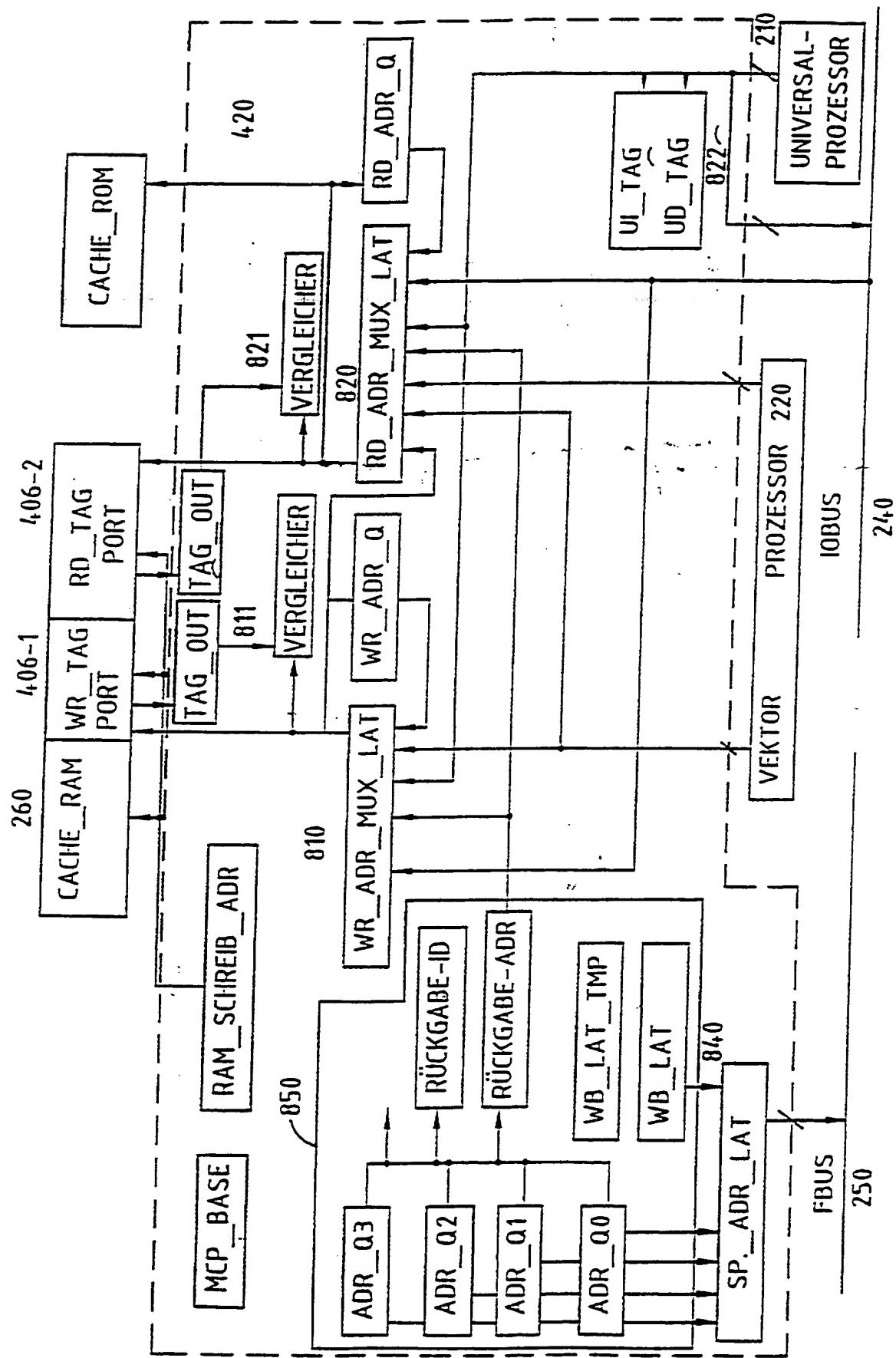


Fig. 8

0	4 MB	internes uROM	510
	4 MB	internes SRAM	520
	64 MB	Lokaler DRAM - Speicher	530
	5 MB	interne FBUS-Einrichtung DRAM-Steuerung VFB-Steuerung DMA-Steuerung KS0122 CODEC serielle Schnittstelle KS0119 CODEC serielle Schnittstelle AD1843 CODEC serielle Schnittstelle	540
	47 MB	Reserviert	550
128 MB	4 MB	interne I/O BUS-Einrichtung Bit-Strom-Prozessor 8259 Interrupt-Steuerung 8254 Zeitgeber 16450 UART Serielle Leitung MSP-Steuer-Register	560
		Reserviert	570
2 GB		andere Hosteinrichtungen (2 GB) (abgebildete auf Host: Adressen von 0 bis 2 GB)	580
4 GB			

Fig. 5

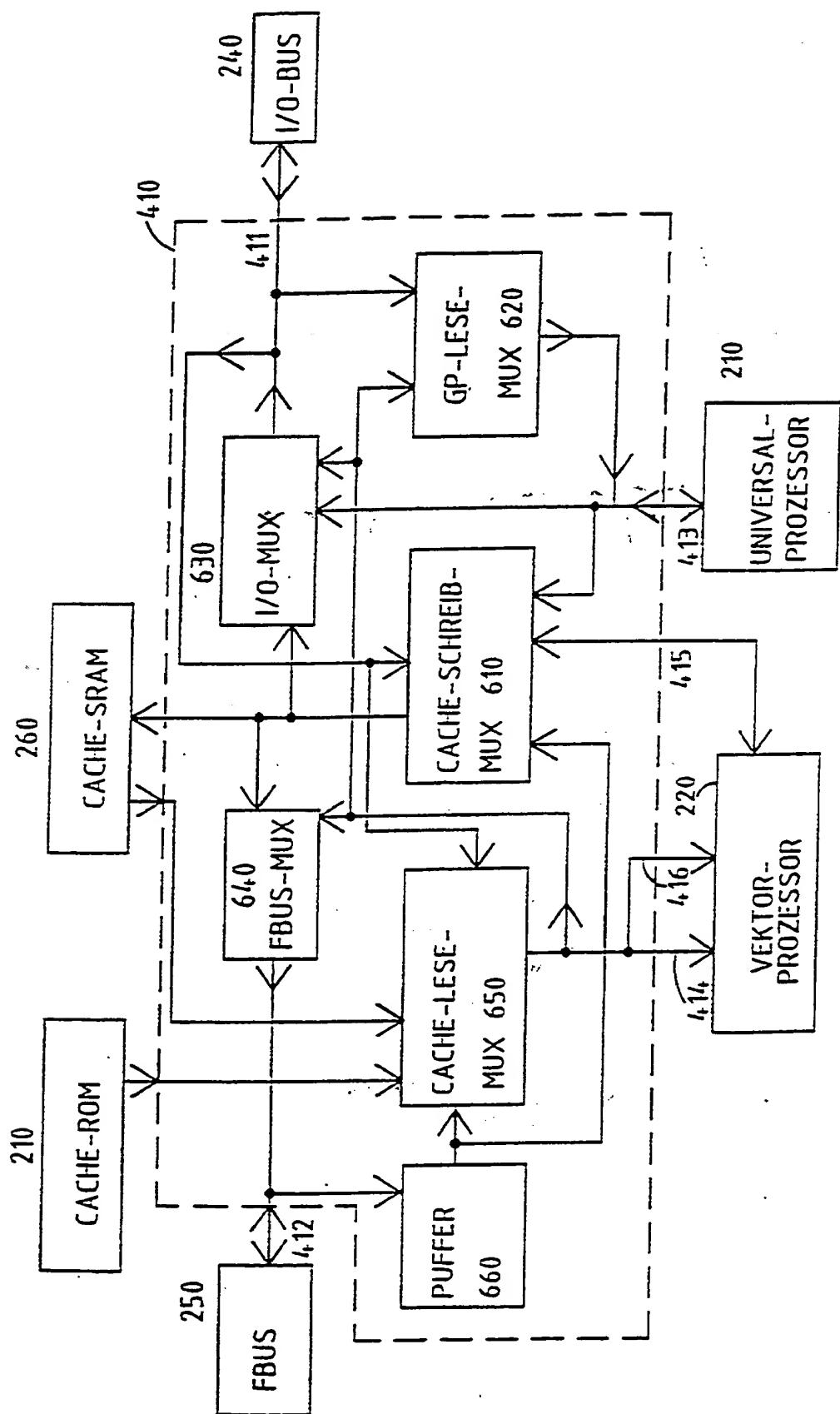


Fig. 6